Recitation 8 — Midterm Feedback and Homework 6

Parallel and Sequential Data Structures and Algorithms, 15-210 (Spring 2013)

March 6, 2013

1 Topics

- Homework 6 is out. You will be required to make some modifications to Dijkstra's Algorithm and also use DFS to find bridges in graphs.
- The first midterm has been graded. We will be covering common mistakes.
- Questions? Comments?

2 Homework 6

For the first part of Homework 6 you will use depth-first search (DFS) numbering to find bridges in graphs. DFS numbering is the main topic of today's recitation.

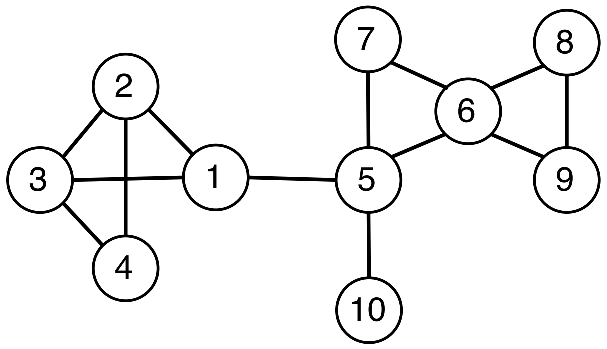
For the second part of Homework 6 you will write three variations of Dijkstra's algorithm:

- Dijkstra's algorithm with multiple source vertices.
- Dijkstra's algorithm with multiple target vertices.
- Dijkstra's algorithm with an additional distance metric used when selecting the next vertex to search. This is known as the A^* heuristic.

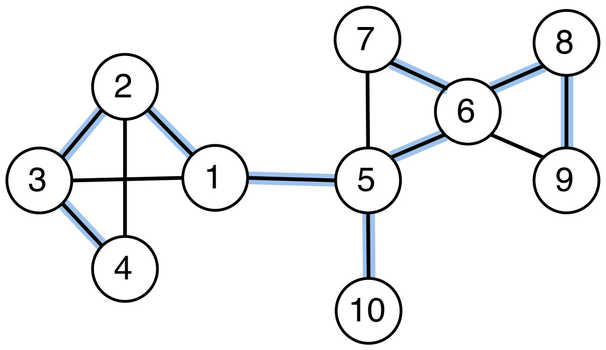
A^* is the most interesting (and practical!) part of this homework and deserves a second mention. As stated above, when selecting the next vertex to add to the set of visited vertices a distance metric is used to estimate which candidates are closer to the target. An intuitive example would be a search across the Euclidean plane with nodes representing points and edge weights representing distances. The estimation metric would just be the straight-line distance between a given node and the target. This metric will make Dijkstra's algorithm prioritize nodes that are close to the goal over those that are far away.

3 DFS

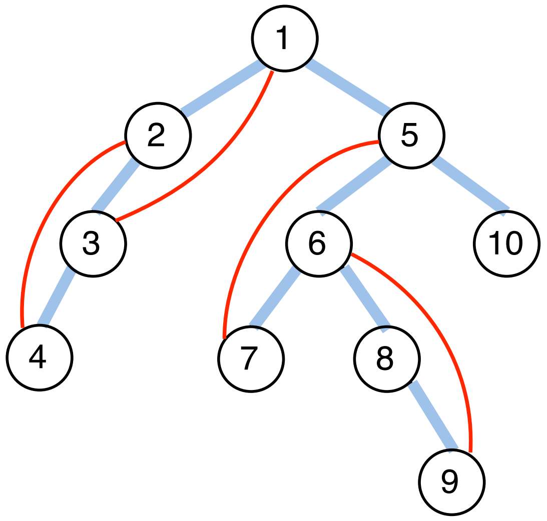
When you apply DFS to a graph, it implicitly defines a spanning tree rooted at the start vertex. If more than one tree is needed to span all vertices in the graph, then we call it a *DFS forest*, although typically we just refer to it as the DFS tree. Edges in the graph that correspond to edges in the DFS tree are called *tree edges*. Edges in the graph that go from a vertex v to an ancestor u in the DFS tree are called *back edges*.



Show the DFS tree on the above graph. Use the vertices in increasing order.



Redraw the tree to illustrate the back edges.



Q: Are all edges in undirected graphs either tree edges and back edges of a DFS tree?
A: Yes.
Notice that the cycle detection algorithm in the lecture notes, simply checks whether the vertex has been visited before. Why is this sufficient? Does it find all cycles?

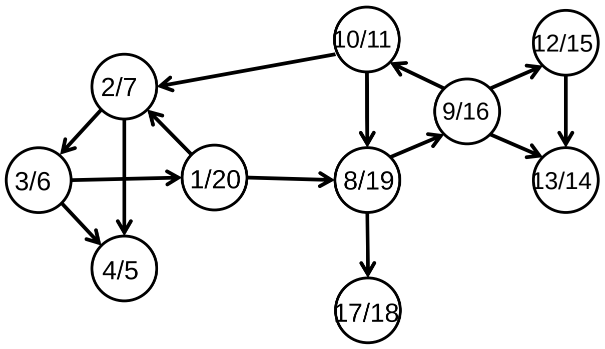
In the homework, you are to find all bridges in an undirected graph. A *bridge* of a graph is an edge whose removal disconnects the graph. In other words, a bridge is not on a cycle.
Q: Which edges are bridges in the above graph? and in the DFS tree?

Q: Are all edges in directed graphs either tree edges and back edges of a DFS tree?
A: No. There can be *forward edges* that are non-tree edges that go from a vertex *v* to a descendant *u* in DFS tree, and *cross-edges* that are all other edges (ie, cross between two subtrees of the DFS tree)

3.1 DFS Numbering

DFS can easily be modified to record the *discovery time* $d[u]$ and *finishing time* $f[u]$ for every vertex u. The discovery time corresponds to the time when a vertex is first visited and the finishing time corresponds to the time when the vertex is last visited.

Here are the discovery time/finishing time numbers for an example directed graph.

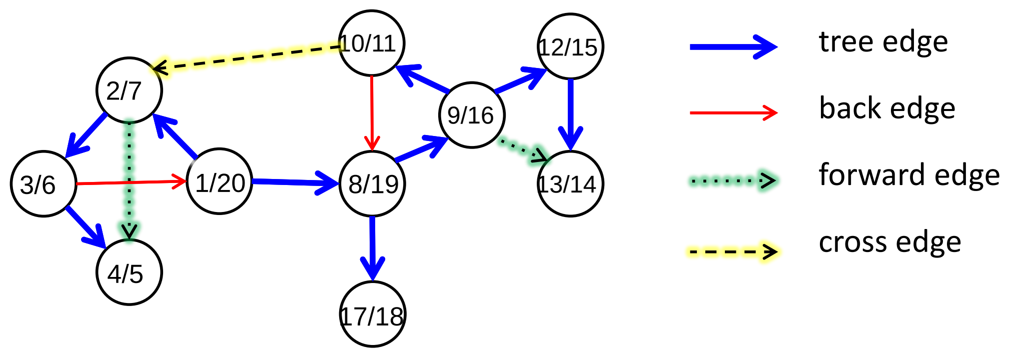


Theorem 3.1 (Theorem 22.7 in Cormen, Leiserson, Rivest and Stein, Introduction to Algorithms)**.** *For any two vertices u and v, exactly one of the three conditions holds:*

1. *the intervals $[d[u], f[u]]$ and $[d[v], f[v]]$ are entirely disjoint, and neither u nor v is a descendant of the other in the DFS forest.*
2. *the interval $[d[u], f[u]]$ is contained entirely within the interval $[d[v], f[v]]$, and u is a descendant of v in a DFS tree.*
3. *the interval $[d[v], f[v]]$ is contained entirely within the interval $[d[u], f[u]]$, and v is a descendant of u in a DFS tree.*

Classify the edges into tree, forward, back or cross edges.

The tree edges of the example graph are shown in blue, forward edges in green, back edges in red and cross edges in yellow.



Q: How would you use the discovery times and finishing times to classify edges into tree, forward, back or cross edges?

A: An edge (u, v) is

1. a tree/forward edge if and only if $d[u] < d[v] < f[v] < f[u]$,
2. a back edge if and only if $d[v] < d[u] < f[u] < f[v]$,
3. a cross edge if and only if $d[v] < f[v] < d[u] < f[u]$.

Q: How do you know if there is a cycle in a graph by using DFS numberings?

3.2 Homework 6 — Bridges

Q: Find a way to use the DFS numbering to identify bridges in a graph. Remember that we are working with undirected graphs for this problem.

4 Exam feedback

The average and median on the first midterm were 71.6 and 72.0, respectively, out of 120. As many of you have guessed, this exam was pretty tough.